US-PAT-NO: 5900870

DOCUMENT-IDENTIFIER: US 5900870 A

TITLE: Object-oriented computer user interface

---------- KWIC ---------

Application Filing Date - AD (1):
19941109

Brief Summary Text - BSTX (50):
Forms Manager

Detailed Description Text - DETX (128):
The heart of Object Lens is the Object Manager. Currently built on top of LOOPS, the Object Manager is responsible for keeping track of all classes and class-instances and their links to each other. It also keeps track of the current state of each object and helps the objects handle messages which they receive by providing support functions for their methods. The Object Manager provides the Forms Manager with the information it needs to present a form. The Object Manger also handles saving and loading objects from permanent storage in the database. In the future, the Object Manager will work with a shared database to do object locking and version control.

Detailed Description Text - DETX (129):
Forms Manager

Detailed Description Text - DETX (130):
When an object receives a "Present Yourself Using Form X" message, it executes the appropriate presentation method by calling the Forms Manager. The Forms Manager also handles user-events addressed to displayed forms and, where necessary, translates them into messages addressed to the appropriate object(s). Examples of messages include:

Detailed Description Text - DETX (131):
The forms manipulated by the Forms Manager are built on top of a set of complex human-interface toolkit objects, which are, in turn, built on top of a window system. While the current implementation of Object Lens uses the toolkit procedures available on the Xerox Interlisp Workstation, the concepts are intended to be portable to other environments, such as X-Windows.

Detailed Description Text - DETX (147):
The control area is at the top of the form. The upper part of the control

area is where the Form Manager places notices and user prompts and is also used to move and re-size the form's window.

Detailed Description Text - DETX (148):
The lower part of the control area has "buttons" for commands which affect either the form itself or the object(s) shown in the data area(s) of the form. It also has a button called "*Other*" which pops-up a menu of less frequently used commands. Forms maintain ordered lists of commands. The one of these lists which is "active" is used to determine what commands will be available: The Form Manager puts as many of the items on the active list onto the button region as will fit, given the window size. The rest are put into the *Other* menu.

Detailed Description Text - DETX (149):
When an object receives a request that it "show" itself, it first determines what form it should be presented through. (Note: the generalized method of using forms as described here is not yet fully implemented in our prototype system.) If an appropriate form was not specified in the message, the object picks its default form. The object then asks the object manager to create or re-use an existing form-instance of the appropriate class. The object then sends a *Show message with all the appropriate object-data to the form instance. From then until an Update Object message is sent to the form, the user interacts with the data in the form and not with the original object. The form determines which list is active by looking at the value of one or more fields in the object to be displayed. This allows the form manager to present different command options depending on the value of status attributes of the object being displayed. A form may have zero, one, or more data areas. At the top of each data area is a label; below the label is where the data about the instance is presented. Form, like other classes, may be sub-classed; many different sub-classes of Form are described below. Currently, the user cannot create Form sub-classes. However, this will change in the future.

Detailed Description Text - DETX (194):
One possible such language which we have partially specified but not yet implemented is called Object Action Language (OAL). An OAL expression consists of a message, a target object, and the arguments which the message takes. Any user-method for any object may be used as part of an OAL expression. If an expression returns a result (e.g. Create Object returns a pointer to that object), that result is placed in a special variable called "The Result." An OAL expression may also call other actions by sending them an execute message. The THEN field of a rule can also contain OAL expressions which are evaluated when the rule fires. Actions are objects which contain procedures which are executed when they receive a Execute message. The message may be sent to the Action either as a result of a rule's firing or by the Forms Manager as a result of a command's being executed. Procedures are written in terms of a limited set of primitives, such as:

Detailed Description Text - DETX (218):
It is useful to distinguish between actions and commands. A command is an instruction given through a form. The instruction may be invoked through the

button-bar or through a pop-up menu. Commands may apply to a window, a form, or to one or more objects. Window commands, such as Close, Move, and Shape, are typically accessed through a pop-up menu, are handled directly by the Window System. Form commands, such as Change Display Format, or Edit Fields to Show, or Select are typically accessed through the button bar and are handled by the Forms Manager. Object commands, such as Show, Add Link, Delete, Send, Answer, and Forward can be accessed either through the button bar (when the object of the action has been selected in advance) or through pop-up menus (when the menu is gotten by clicking-middle on the object of the action). Object actions are also handled by the Forms Manager, which translates them into action requests to the Action Exec.

Detailed Description Text - DETX (219):

The Delete Selection command is a useful example. After the user has issued a series of commands to the form in order to select a set of objects contained in a folder and being shown by the form, the user goes to the button-bar and chooses Delete Selection. The Forms Manager uses its list of selected objects to translate this command into a series of Delete actions which changes the contents of the folder to remove the selected items. Like actions, commands are also currently hard-coded. Also like actions and forms, in the future, users will be able to write commands and incorporate them into forms.